Kernel Bandwidth Selection via SGD

Gilberto Garcia Pérez
gilberto.gape@gmail.com

Andrés Potapczynski[*]
ap3635@columbia.edu

Abstract

We propose an algorithm to select the kernel bandwidths that is faster when compare to the standards approaches and is scalable in the number of variables. The standard methods select the bandwidth by using derivative free optimization and rely on computing the generalization error based on LOO. This optimization technique does not scale to the number of variables and the generalization error based on LOO is highly biased. We circumvent both problems by posing an optimization problem that computes a better generalization error and that uses the gradient to efficiently explore the parameter space.

1 Introduction

- Check exhaustively that the idea has not been developed elsewhere.
- Revise the kernel regression literature to see what it the most novel technique
- Expand Motivation from abstract.
- Explain how AD makes this idea possible and computationally feasible Abadi and et. al. [2015]
- Mention that the code is available in GitHub

2 Mathematical Formulation

The following idea has been explored in Maclaurin et al. [2015] and can be applied to any Machine Learning model easily. Let $\omega \in \Omega$ denote the H hyperparameters of a given model. We reparameterize each ω_j as a function of ξ_j for $j = 1, \ldots, H$ where $\xi \in \mathbb{R}^H$. Thus, we want to minimize

$$F(\xi) = L(y^{test}, \hat{y}(x^{test}, \xi, \mathfrak{D}^{train}))$$

where $L(\cdot, \cdot)$ is a loss function that estimates the generalization error of the model, y^{test} is the test data being used for the estimation and $\hat{y}$ are the predictions that depend on both the hyperparameter choice ξ and the training data $\mathfrak{D}^{train}$. Thus, we use AD to numerically compute

$$\nabla F(\xi) = \nabla L(y^{test}, \hat{y}(x^{test}, \xi, \mathfrak{D}^{train}))\nabla \hat{y}(x^{test}, \xi, \mathfrak{D}^{train})$$

and hence enable an updating schema of the form

$$\xi^{t+1} = \xi^t - \rho \nabla F(\xi^t)$$

where t denotes the iteration number and ρ the step-size. As implicitly stated above, we need that $\nabla \hat{y}(\cdot, \cdot, \cdot)$ and $L(\cdot, \cdot)$ be differentiable functions on the appropriate argument. The choice of $L(\cdot, \cdot)$ used in this paper is the usual M-fold CV MSE estimator.

[*] ...

Under discussion.

$$L(y^{test}, \hat{y}(\xi, \mathfrak{D}^{train})) = \frac{1}{M} \sum_{j}^{M} \sum_{i}^{N_j} (y_i^{test_j} - \hat{y}_i(x_i^{test_j}, \xi, \mathfrak{D}^{train_j}))^2$$

where N_j denotes the number of test observations in fold j for $j = 1, \ldots, M$, $y_i^{test_j}$ is the i-th label/output for fold j, similarly $x_i^{test_j}$ is the i-th test observation for fold j, $\mathfrak{D}^{train_j}$ the training data available in fold j and, finally, $\hat{y}_i(x_i^{test_j}, \xi, \mathfrak{D}^{train_j})$ is the model's prediction. As illustration, we show the functional forms for the case of Kernel Regression,

$$\hat{y}_i(x_i^{test_j}, \xi, \mathfrak{D}^{train_j}) = \sum_{r}^{N_j^{train}} y_r^{train_j} \frac{K_h(x_i^{test_j} - x_r^{train_j})}{\sum_k K_h(x_i^{test_j} - x_k^{train_j})}$$

where

$$K_h(z) = \exp\left(-\frac{1}{2} \sum_{d}^{D} \left(\frac{z_d}{h_d}\right)^2\right)$$

and the bandwidth is again re-parameterized positively as $h = \exp(\xi)$

3 Experiments

3.1 Kernel Regression

For our toy simulation, we used a function of two variables where having a different bandwidth for each would be advantageous. For that, we set a linear dependence on the first variable and a non-linear on the second. Moreover, we added interactions effects to increase the difficulty of the problem. Therefore, after sampling $x_i^{(s)} \sim U[0, 10]^2$ and $\epsilon_i^{(s)} \sim N(0, 0.5)$ we generated

$$y_i^{(s)} = x_{i,0}^{(s)} - \left(x_{i,1}^{(s)}\right)^2 + 0.3 \log\left(1 + x_{i,0}^{(s)} \cdot x_{i,1}^{(s)}\right) + \epsilon_i^{(s)}$$

for $i = 1, \ldots, N$ and $s = 1, \ldots, S$ where N is the number of observations for each sample and S the total number of samples. In figure 1 we summarize the results. Similar to the previous experiment, we see on the right plot that the error decreases rapidly and almost at 20 iterations all the simulations have converged. Additionally, on the plot of the right we exhibit the qualitative improvement that having two bandwidths creates by having a much smoother approximation in the end of the range.

As for the real-life data set example, we again used an example of Hastie et al. [2009] but now from its kernel Regression chapter. We used the South African Heart Disease data. **[...]**

Shalizi [2019]

4 Conclusions

[...]

5 Future Work

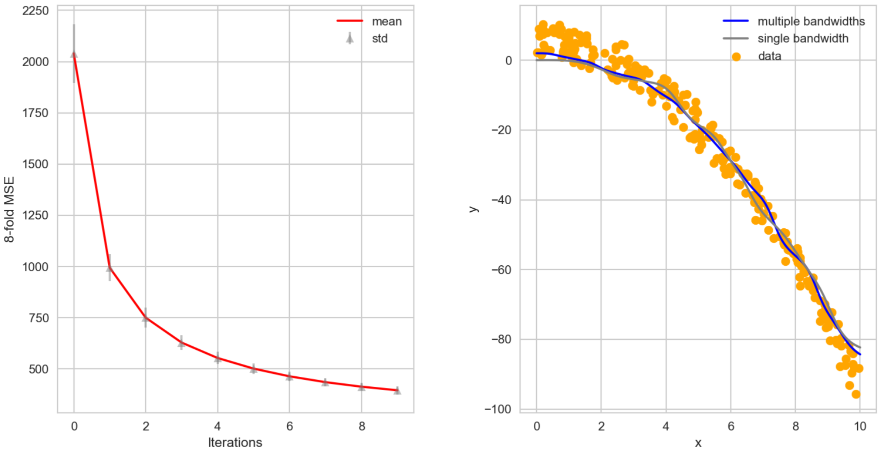
[...]

References

M. Abadi and et. al. Tensorflow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`.

T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2009.

Figure 1: Kernel Regression Simulation Results.



D. Maclaurin, D. Duvenaud, and R. P. Adams. Gradient-based hyperparameter optimization through reversible learning. *arXiv:1502.03492v3*, 2015.

C. R. Shalizi. Advance data analysis from an elementary point of view, 2019. URL `https://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/`.